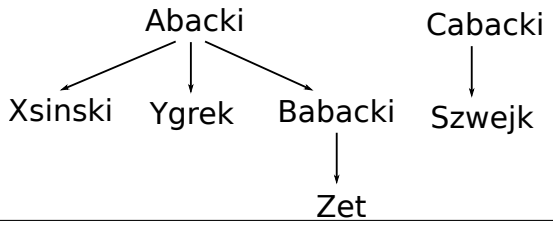
Imię i nazwisko: **22Z kol1**

Proszę wpisywać odpowiedzi w miejscach na to przeznaczonych

Zadanie 1 - obiektowe wzorce projektowe (2pkt)

Obiekty 'Soldier' i 'Commander' opisują odpowiednio żołnierza i dowódcę. Niestety, nie można reprezentować drzewistych struktur obiektów, przykład takiej sytuacji jest pokazany obok. Na rysunku Abacki, Babacki i Cabacki to dowódcy. Popraw kod, aby można było reprezentować sytuacje, gdy podległym dowódcy jest osoba, która kimś dowodzi.




```
using PtrCreature = unique_ptr<Creature>;

class Creature {
public:
    virtual ~Creature() = 0;
    virtual PtrCreature clone() const = 0;
};

class Soldier : public Creature {
public:
    Soldier(const string& name) : name_(name) {}
    Soldier(const Soldier& s) : name_(s.getName()) {}
    PtrCreature clone() const override { return PtrCreature(new Soldier(*this)); }
    const string& getName() const { return name_; }
private:
    string name_;
};

class Commander : public Creature {
public:
    Commander(const string& name) : name_(name) {}
    Commander(const Commander& cmd) : name_(cmd.getName()) {
        for(const PtrCreature& soldier : cmd.subordinates_ ) { //kopiuje osoby podlegle
            subordinates_.push_back( soldier->clone() );
        }
    }
    PtrCreature clone() const override { return PtrCreature(new Commander(*this)); }
    const string& getName() const { return name_; }
    virtual void addSubordinate(const string& name) { subordinates_.push_back(PtrCreature(new Soldier(name))); }
    const vector<PtrCreature>& getSubordinates() const { return subordinates_; }
private:
    string name_;
    vector<PtrCreature> subordinates_; //podlegli
};
```

Uwagi do prowadzącego (R. Nowaka):

Zadanie 2 - obiektowe wzorce projektowe (2pkt)

Dla klas 'Soldier' i 'Commander', pokazanych w poprzednim zadaniu dostarczono wzorzec wizytatora, tzn. dodano klasę Visitor oraz metody accept, tak jak pokazano poniżej.

```
class Visitor {
public:
    virtual void visit(const Soldier& s) = 0;
    virtual void visit(const Commander& c) = 0;
    virtual ~Visitor() {}
};
class Soldier : public Creature {
    //...
    virtual void accept(Visitor& v) const { v.visit(*this); }
    //...
};
class Commander : public Creature {
    //...
    virtual void accept(Visitor& v) const { v.visit(*this); }
    //...
};
```

Zaimplementuj metodę 'countSoldiers', która zwraca ilość żołnierzy podległych (bezpośrednio lub pośrednio) osobom, reprezentowanym w danej kolekcji 'v'. Na przykład, dla kolekcji 2 elementowej zawierającej 'Abackiego' i 'Cabackiego', gdy obiekty są powiązane tak jak na rysunku w Zadaniu 1, funkcja powinna zwrócić 4.

```
int countSoldiers(const vector<PtrCreature>& v) {
```

Zadanie 3 - cykl życia oprogramowania (2pkt)

Proszę odpowiedzieć na następujące pytania:

1. Kiedy można powiedzieć o wymaganiu, że jest atomowe? Podaj przykład – własny, wskazujący na zrozumienie tematu.

2. W którym momencie procesu testowania uzyskujemy gwarancję poprawności programu? Kiedy możemy go dostarczyć klientowi?

Zadanie 4 - SOLID (2pkt)

Czy fragment kodu pokazany niżej jest zgodny z regułami SOLID? Jeśli nie, to w jaki sposób je narusza (może naruszać więcej niż jedną regułę)?

Popraw kod.

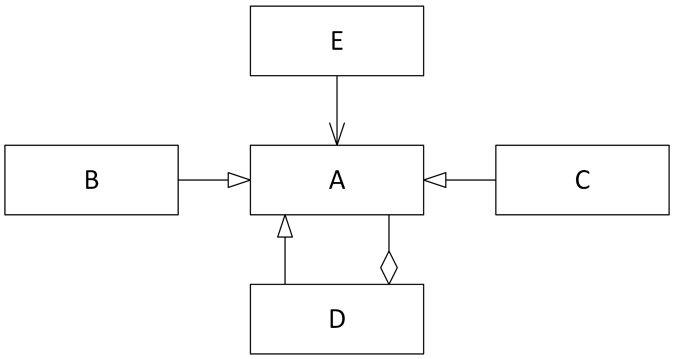
```
class Page
{
public:
  const std::vector<std::string>& words() const;

  bool isSpellingCorrect() const
  {
    EnglishDictionary dictionary;
    return std::ranges::all_of(words(), [&dictionary](const auto& w) { return dictionary.hasElement(w); });
  }

  void save(const std::string& fileName)
  {
    std::ofstream out(fileName);
    std::ranges::copy(words(), std::ostream_iterator<std::string>(out, "␣"));
  }
};
```

Zadanie 5 - UML (2pkt)

Opisz wszystkie relacje jakie widzisz między klasami widocznymi na diagramie. Jeśli diagram przypomina znajomy wzorzec, zaproponuj ciekawsze nazwy klas.



Uwagi do prowadzącego (K. Grochowskiego):